Supplemental Material for the Manuscript Aberration-Aware Depth-from-Focus

Xinge Yang, *Student Member, IEEE,* and Qiang Fu, and Mohamed Elhoseiny, *Member, IEEE,* and Wolfgang Hiedrich, *Fellow, IEEE*

Abstract—Computer vision methods for depth estimation usually use simple camera models with idealized optics. For modern machine learning approaches, this creates an issue when attempting to train deep networks with simulated data, especially for focus-sensitive tasks like Depth-from-Focus. In this work, we investigate the domain gap caused by off-axis aberrations that will affect the decision of the best-focused frame in a focal stack. We then explore bridging this domain gap through aberration-aware training (AAT). Our approach involves a lightweight network that models lens aberrations at different positions and focus distances, which is then integrated into the conventional network training pipeline. We evaluate the generality of network models on both synthetic and real-world data. The experimental results demonstrate that the proposed AAT scheme can improve depth estimation accuracy without fine-tuning the model for different datasets. The code and models will be made publicly available.

Index Terms—Depth from Focus, Optical Aberration, Ray Tracing, Point Spread Function

1 Dataset

1.1 Image Dataset

Our experiments involve two types of focal stacks: simulated focal stacks generated using RGBD images and real-world captured focal stacks. Simulated focal stacks comprise both synthetic scenes and real-world scenes, while real-world captured focal stacks consist of both outdoor scenes and indoor scenes. To facilitate aberration-aware training, we require access to lens data, focus distances, focal stacks, and depth maps. Unfortunately, no such dataset is currently available, so we created our own training data based on simulated focal stacks, which allowed us to leverage known lens models and select appropriate focus distances. For testing, we used both simulated focal stacks and real-world captured focal stacks.

To generate simulated focal stacks using RGBD data, we select the FlyingThings3D [1], [2], Middlebury2014 [3], and Matterport3D [4] datasets, which provide RGBD images. To simulate the focal stacks, we use the normalized pixel coordinates, focus distance, and depth map to calculate the point spread function (PSF) for each pixel, and then perform a convolution to generate focused images. The focus distances are chosen linearly within the minimum and maximum depth range of each image to mimic the focal swapping process. In addition to regular data augmentation techniques, such as random flips and color jittering, we also introduce a random perturbation to each focus distance to enhance the variety of our simulated focal stacks. An example of our simulated focal stacks can be seen in Fig. 1.

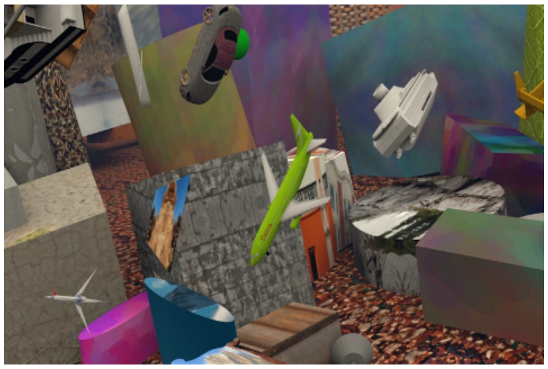
To capture our real-world captured focal stacks, we position the camera and focus it on different objects while measuring the distance from the focused objects to the camera (for the indoor scene), or reading the focus distances from the EXIF metadata of the photos (for the outdoor scenes). Our real-world focal stacks comprise 24 outdoor scenes and 1 indoor scene. Fig. 2 depicts the experimental setup for the indoor scene. We use the LiDAR sensor on an iPhone 14 pro to scan the 3D scene and load the scanned data into Blender software. We then adjust the camera position and view angle to align with the focused images. Once aligned, we render the depth map with Blender and use it as the ground truth for quantitative evaluation. Due to the breathing effect, the focused images are not well-aligned. We load them into Photoshop, align them, and then output the aligned focal stack.

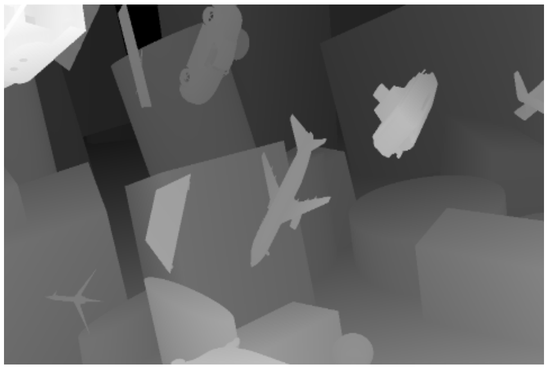
The resolution ratio of training and testing images is determined by the camera sensor resolution, and any resizing operation can alter the aberration property of the lens. However, downsampling is feasible due to the physical properties of the PSF. Although optical aberrations are more pronounced at higher image resolutions, network training processes cannot handle the extremely high-resolution images produced by commercial cameras with megapixel sensors. Therefore, we find a balance between performance and memory consumption that allows us to observe aberrations in the image without using excessive memory during network training. In our experiments, we used two different image resolutions: 480×640 for the 50mm F/2.8 lens and 640×960 for the Canon RF 50mm F/1.8 lens. As shown in Fig.3 (main paper) and Fig. 4, the off-axis aberrations can be easily seen at the resolutions we use.

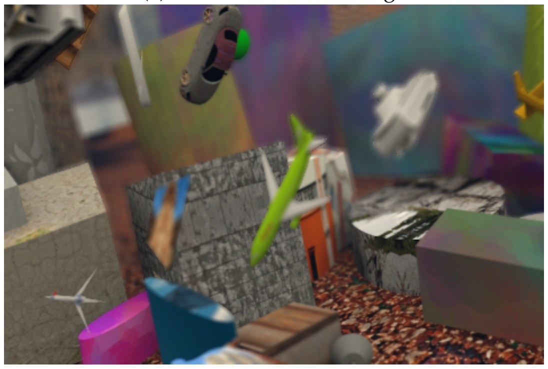
1.2 Lens Data

Our experiments involved two lenses: the 50mm F/2.8 lens, generated using lensnet.com [5], and the Canon RF 50mm

• *W. Heidrich is with the Department of Computer Science and Electrical and Computer Engineering, King Abdullah University of Science and Technology, Saudi Arabia, 23955.*
E-mail: wolfgang.heidrich@kaust.edu.sa
• *X. Yang, Q. Fu, M. Elhoseiny is with King Abdullah University of Science and Technology.*


(a) All-in-focus RGB image


(b) Ground-truth depth map


(c) Focused image example 1

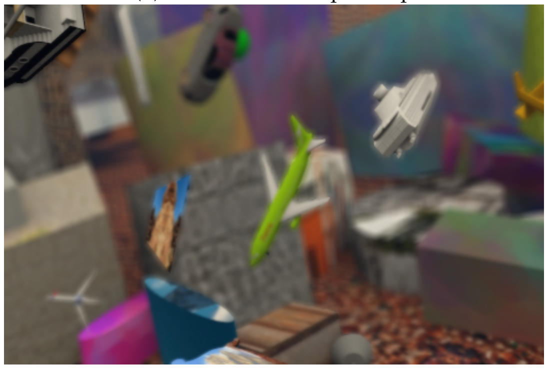
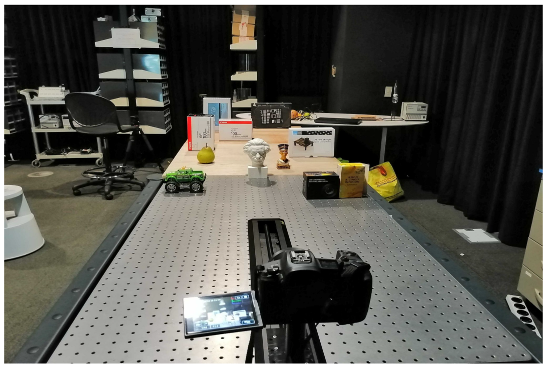
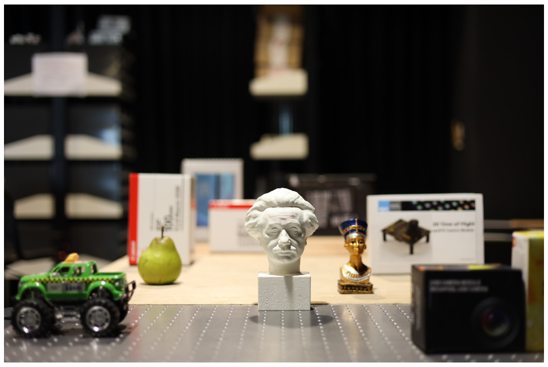
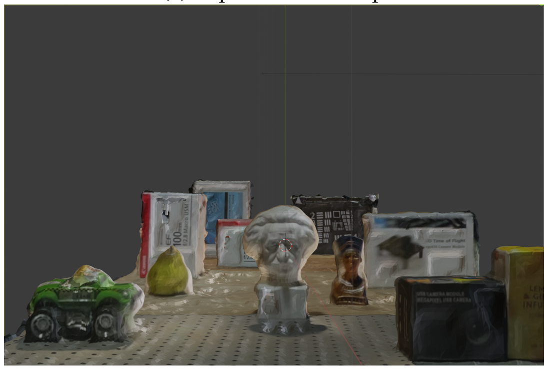

(d) Focused image example 2

Fig. 1. Simulated focal stacks with RGBD images.


(a) Experimental setup


(b) Focused image example


(c) LiDAR scan result

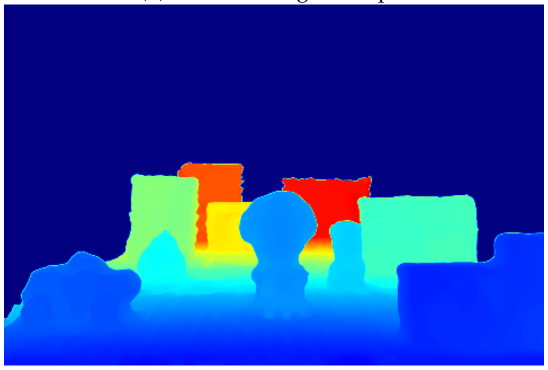

(d) Rendered depth map

Fig. 2. Real-world indoor scene. (a) We set up an indoor scene with objects placed at different positions. (b) We focus the camera on different objects and capture focal stacks. (c) We use the LiDAR sensor on iPhone 14 pro/iPad pro to scan the 3D scene and load the scanned data into Blender. We adjust the camera position and view angle to align with the focused image. (d) We render the depth map with Blender and use it as the ground truth for quantitative evaluation.

TABLE 1
Lens data for the 50mm F/2.8 lens used in the paper. The lens is generated by [5].

Surface	Radius [mm]	Thickness [mm]	Material (n/V)	Semi-diameter [mm]	Conic	α_4	α_6	α_8	α_{10}	α_{12}
1 (Sphere)	25.445	5.120	1.7290/54.494	15.00	0	0	0	0	0	0
2 (Sphere)	90.909	0.847		15.00	0	0	0	0	0	0
3 (Sphere)	22.124	2.937	1.6517/58.5020	12.50	0	0	0	0	0	0
4 (Sphere)	40.000	1.731		12.50	0	0	0	0	0	0
5 (Sphere)	204.082	1.782	1.6990/30.1789	12.50	0	0	0	0	0	0
6 (Sphere)	16.556	5.183		10.00	0	0	0	0	0	0
7 (Aper)		1.414		9.00	0	0	0	0	0	0
8 (Sphere)	-36.101	6.749	1.6204/60.3100	10.00	0	0	0	0	0	0
9 (Sphere)	-31.546	0.127		12.50	0	0	0	0	0	0
10 (Sphere)	44.643	7.151	1.7551/52.2945	15.00	0	0	0	0	0	0
11 (Sphere)	769.231	29.792		15.00	0	0	0	0	0	0
Sensor				21.63						

TABLE 2
Lens data for the Canon RF 50mm F/1.8 lens used in the paper.

Surface	Radius [mm]	Thickness [mm]	Material (n/V)	Semi-diameter [mm]	Conic	α_4	α_6	α_8	α_{10}	α_{12}
1 (Sphere)	28.621	4.20	N-LASF41	15.00	0	0	0	0	0	0
2 (Sphere)	68.136	0.18		14.24	0	0	0	0	0	0
3 (Sphere)	17.772	6.70	N-LASF43	12.45	0	0	0	0	0	0
4 (Sphere)	59.525	1.10	SF6	10.89	0	0	0	0	0	0
5 (Sphere)	11.427	5.27		8.89	0	0	0	0	0	0
6 (Aper)		6.20		7.50	0	0	0	0	0	0
7 (Sphere)	-16.726	0.90	SF5	7.48	0	0	0	0	0	0
8 (Sphere)	-29.829	0.83		7.73	0	0	0	0	0	0
9 (ASphere)	-25.000	2.95	K5G20	7.76	0	-4.1203e-5	-2.9002e-7	-4.6712e-9	7.9065e-11	-9.2847e-13
10 (ASphere)	-18.373	0.98		9.07	0	-2.4162e-5	-3.2915e-7	1.9110e-10	-9.2859e-13	-2.2919e-13
11 (Sphere)	280.004	4.60	N-LAK10	12.22	0	0	0	0	0	0
12 (Sphere)	-34.002	25.67		12.86	0	0	0	0	0	0
Sensor				21.63						

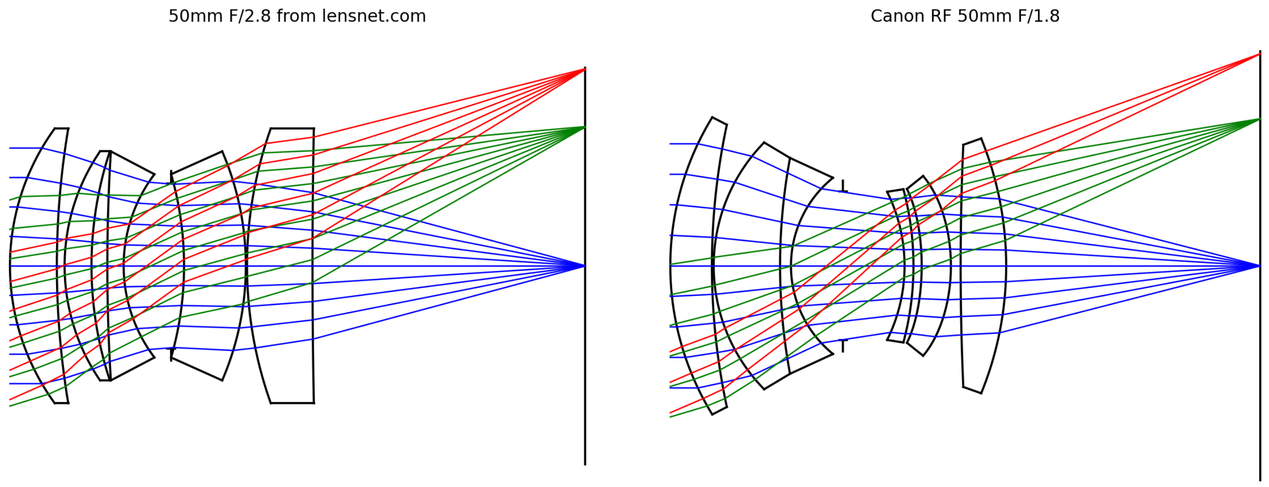


Fig. 3. Lens setup. Left: a 50mm F/2.8 lens from lensnet.com [5]. Right: the Canon RF 50mm F/1.8 lens.

F/1.8 lens, a commercially available lens. The lens data for both lenses is presented in Table. 1 and Table.2, respectively. The 2D lens structures with ray paths are displayed in Fig. 3. Notably, the 50mm F/2.8 lens exhibits poorly corrected optical aberrations, thus the performance improvement with the AAT scheme is more obvious compared to the Canon lens which has well-corrected optical aberrations.

2 PSF Network Training and Evaluation

In order to effectively train the PSF network, we needed to sample a sufficient amount of input data to account for the complex optical properties, particularly since the PSF varies significantly with the object position and focus distance. When the focus distance is short, even small changes in distance can produce large variations in the PSF. Additionally, for depths close to the focus distance, the PSF undergoes rapid changes, requiring the sampling of numerous depths in the vicinity of the focus distance.

In our experiments, we utilize the principle of importance sampling to optimize the sampling strategy. Specifically, we sample more focus distances from the near range and more depths in the vicinity of the focus distance. For the sampling of normalized off-axis x and y coordinates, we use a uniform distribution.

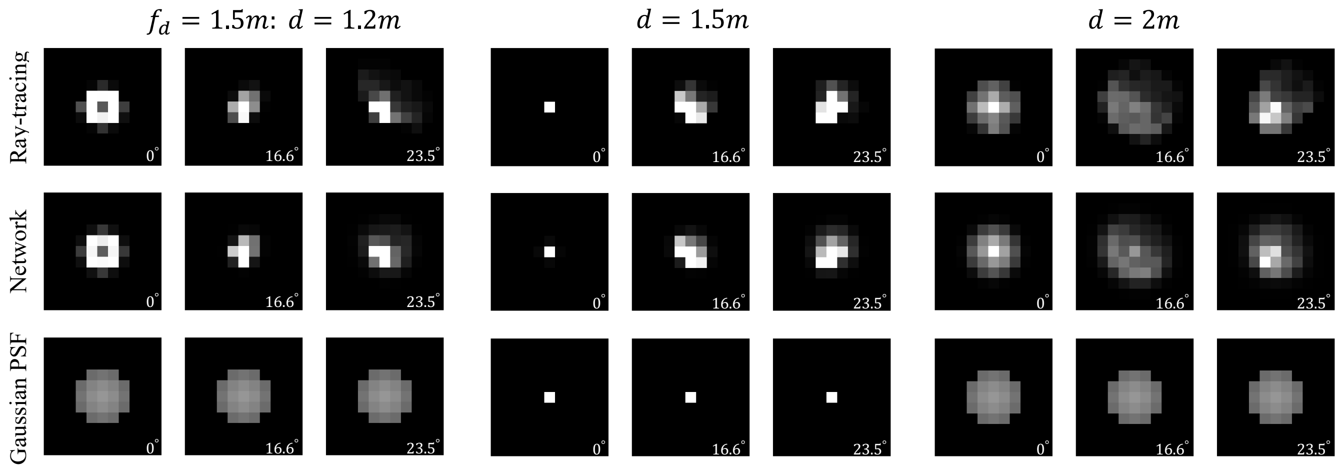


Fig. 4. The PSF evaluation results for the 50mm F/2.8 lens. We focus the lens to a distance of 1.5m and evaluate the PSF of the three methods at three depths and three view angles. Our proposed PSF network produces a PSF that is similar to the ground truth (ray tracing). In contrast, the Gaussian PSF exhibits significant differences, particularly at large view angles. The PSF of a real lens varies with different view angles due to the presence of off-axis optical aberrations, while the Gaussian PSF model neglects this aberration.

For each training iteration, we generate 256 input data with different object positions but the same focus distance. We use the same focus distance within each iteration because changing the focus distance requires adjusting the sensor position of the lens model. However, for different iterations, we use different focus distances. We train the PSF network for 100,000 iterations, which we deem sufficient to cover most situations. Once trained, we freeze the parameters and use the PSF network in subsequent depth-from-focus training to generate photorealistic images.

There are other methods for representing the PSF using neural networks. For example, Tseng et al. [6] propose an architecture with an MLP encoder and a CNN decoder to estimate the PSF grid at given depths. We modify the input to exclude the optics parameters and include focus distance parameters, and train the network for comparison. Our pure-MLP network outperforms the other network in terms of PSF representation accuracy. Moreover, our PSF network is better suited for the DfF problem, as different image pixels have different depths, whereas the other PSF network tends to estimate the PSF grid for an entire depth plane. Thus, we can use our PSF network to simultaneously estimate PSFs for points with different depths, while the other network is harder to implement for this purpose.

3 Depth-from-Focus Network Training and Evaluation

3.1 Local PSF Convolution

Existing methods usually use the same PSF kernel for convolving the entire image. However, in our experiments, we use different PSF kernels for each pixel. This local convolution operation does not have a specific function, thus we provide our implementation in PyTorch, as follows:

```
def local_psf_render(input, psf, kernel_size=11):
""" Blurs image with dynamic Gaussian blur.

Args:
    input (Tensor): The image to be blurred (N, C, H
        , W).
    psf (Tensor): Per pixel local PSFs (1, H, W, ks,
        ks)
    kernel_size (int): Size of the PSFs. Defaults to
        11.

Returns:
    output (Tensor): Rendered image (N, C, H, W)
"""

if len(input.shape) < 4:
    input = input.unsqueeze(0)

b,c,h,w = input.shape
pad = int((kernel_size-1)/2)

# 1. pad the input with replicated values
inp_pad = torch.nn.functional.pad(input, pad=(pad,
    pad,pad,pad), mode='replicate')
# 2. Create a Tensor of varying Gaussian Kernel
kernels = psf.reshape(-1, kernel_size, kernel_size)
kernels_rgb = torch.stack(c*[kernels], 1)
# 3. Unfold input
inp_unf = torch.nn.functional.unfold(inp_pad, (
    kernel_size,kernel_size))
# 4. Multiply kernel with unfolded
x1 = inp_unf.view(b,c,-1,h*w)
x2 = kernels_rgb.view(b, h*w, c, -1).permute(0, 2,
    3, 1)
y = (x1*x2).sum(2)
# 5. Fold and return
return torch.nn.functional.fold(y,(h,w),(1,1))
```

For high-resolution images, we may encounter memory issues. One solution is to perform local PSF convolution on image patches, which can be implemented as follows:

```
def local_psf_render_high_res(input, psf, patch_size
    =[320, 480], kernel_size=11):
    B, C, H, W = input.shape
    img_render = torch.zeros_like(input)
    for pi in range(int(np.ceil(H/patch_size[0]))):
        for pj in range(int(np.ceil(W/patch_size[1])
            )):
            low_i = pi * patch_size[0]
            up_i = min((pi+1)*patch_size[0], H)
            low_j = pj * patch_size[1]
            up_j =  min((pj+1)*patch_size[1], W)

            img_patch = input[:, :, low_i:up_i,
                low_j:up_j]
```

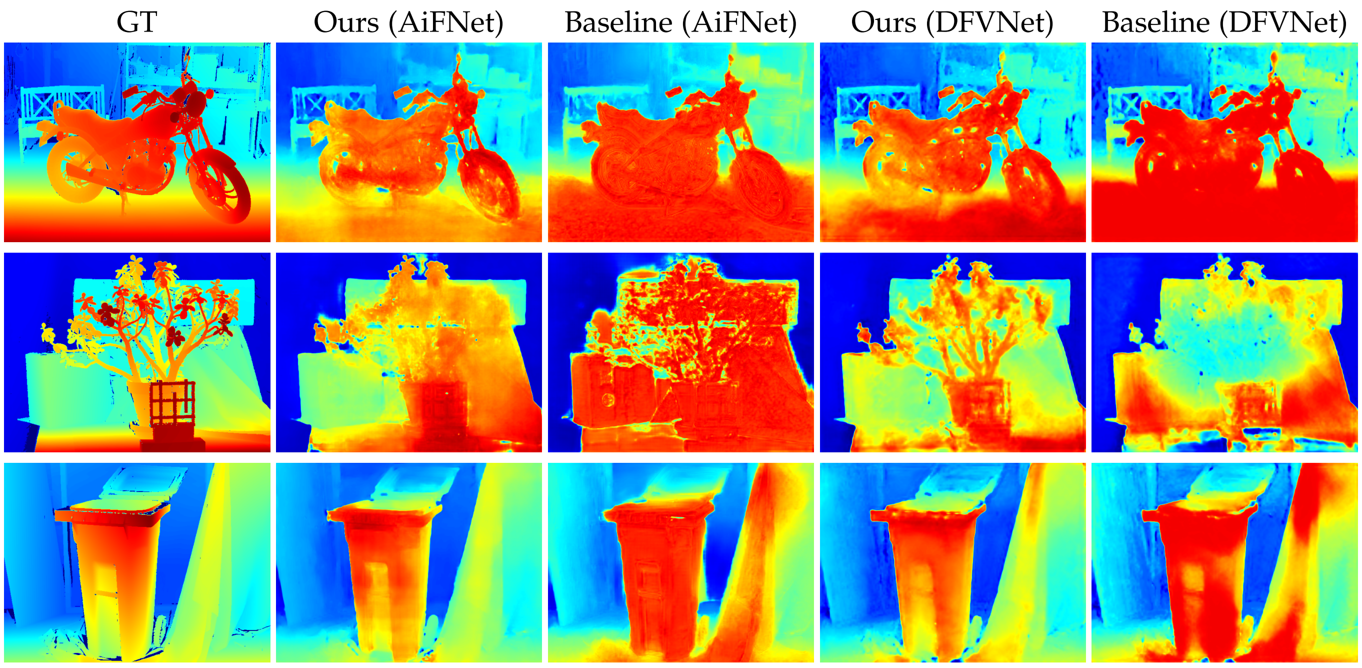


Fig. 5. Qualitative results on simulated focal stacks. With the AAT scheme, both network models predict more accurate and finer depth maps, while the non-AAT models fail to distinguish between adjacent objects and also mispredict the depth of some edge objects.

```
            psf_patch = psf[:, low_i:up_i, low_j:
                up_j, :, :]

            img_render[:, :, low_i:up_i, low_j:up_j]
                 = local_psf_render(img_patch,
                psf_patch, kernel_size=kernel_size)

    return img_render
```

3.2 DfF Training and Evaluation

Simulating focal stacks from RGBD datasets allows us to use desired lens models and flexible focus distances. During the depth-from-focus (DfF) training, we follow the original training procedure and hyperparameter settings as reported in the original papers for the AiFNet [2] and the DFVNet [7]. We find that stacking the focused images in order of focus distance leads to better training results than stacking them randomly. Therefore, we form both training and testing focal stacks following this principle.

For the experiments on simulated focal stacks, we evaluate the generalizability of the models by training them with focal stacks generated on synthetic RGBD images and testing them with focal stacks generated on real-world RGBD images. There is a significant domain gap between semantic information of the training and testing datasets. However, the experimental results show that the domain gap can be bridged effectively using our proposed AAT scheme. Additional evaluation results are presented in Fig.5.

It is worth noting that our results are not contradictory to those in the original papers [2], [7], as the training and testing datasets used in those papers were also simulated with the same lenses (thin lens model of the same focal length and aperture), which is consistent with the conclusion we are making.

References

[1] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 4040–4048. 1

[2] N.-H. Wang, R. Wang, Y.-L. Liu, Y.-H. Huang, Y.-L. Chang, C.-P. Chen, and K. Jou, “Bridging unsupervised and supervised depth from focus via all-in-focus supervision,” in *Int. Conf. Comput. Vis.*, 2021, pp. 12 621–12 631. 1, 5

[3] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, “High-resolution stereo datasets with subpixel-accurate ground truth,” in *German conference on pattern recognition*. Springer, 2014, pp. 31–42. 1

[4] Y. Zhang and T. Funkhouser, “Deep depth completion of a single RGB-D image,” in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 175–185. 1

[5] G. Côté, J.-F. Lalonde, and S. Thibault, “Deep learning-enabled framework for automatic lens design starting point generation,” *Opt. Express*, vol. 29, no. 3, pp. 3841–3854, 2021. 1, 3

[6] E. Tseng, A. Mosleh, F. Mannan, K. St-Arnaud, A. Sharma, Y. Peng, A. Braun, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide, “Differentiable compound optics and processing pipeline optimization for end-to-end camera design,” *ACM Trans. Graph.*, vol. 40, no. 2, pp. 1–19, 2021. 4

[7] F. Yang, X. Huang, and Z. Zhou, “Deep depth from focus with differential focus volume,” in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 12 642–12 651. 5